# DØ Controls and Monitoring

## 1 Introduction

This chapter discusses the control system architecture of the current D0 experiment, how EPICS (Experimental Physics and Industrial Control System [1]) was adapted to meet the control and monitoring requirements of a large, high-energy physics detector, and how a formal control system contributes to the management of detector operations. EPICS, an integrated set of software building blocks for implementing a distributed control system, has been adapted to satisfy the slow controls needs of the D0 detector (1) by extending the support for new device types and an additional field bus, (2) by the addition of a global event reporting system that augments the existing EPICS alarm support, and (3) by the addition of a centralized database with supporting tools for defining the configuration of the control system.

## 2 EPICS and Its Extensions

Following the first running period for the DØ experiment, which ended in 1995, the computing policy of the laboratory decreed that future experiment software must be developed from platform-independent components. Since the DØ controls and monitoring group was small and the period before the beginning of the next running period was short, recasting the existing slow-controls system in the new formalism was not practical.

After a brief survey of the field, the controls group selected EPICS to provide the building blocks for our new controls effort. EPICS uses a distributed client-server architecture consisting of host-level nodes, called clients, that run application programs and Input/Output Controller (IOC) nodes, called servers, that interface directly with the detector hardware. The two classes of nodes are connected by a local area network. Clients access process variable (PV) objects on the servers using the EPICS channel access protocol.

The principal reasons for selecting EPICS were (1) the availability of device interfaces that matched or were similar to our hardware, (2) the ease with which the system could be extended to include our experiment-specific devices, and (3) the existence of a large and enthusiastic user community that understood our problems and were willing to offer advice and guidance.

One of the unique properties of the DØ detector interface is the use of the MIL-STD-1553B serial bus for all control and monitoring operations of the detector and of the electronics components located in the remote collision hall. Since the detector is inaccessible for extended periods of time, a robust, high-reliability communication field bus is essential. We extended EPICS by providing a queuing driver for MIL-STD-1553B controllers and a set of device support routines that provided the adaptive interface between the driver and the standard EPICS PV support records. Once these elements were in place, all of the features of EPICS were available for use with our remote devices.

High voltage (HV) channel control is an example of extending the basic PV record support [2]. In this case, building a compound device from individual PV records was not feasible because of the complexity of the HV device and the speed requirements. A generic HV record support module was developed based upon the extended Harel state machine model [3]. The record support module provides the required, high-level behavior: (1) linear ramping with parabolic end sections, (2) retries for convergence, (3) trip condition recovery, and (4) limits control. Device support modules then adapt the HV record to specific hardware devices. Although developed for a specific device, the record support is non-device specific and may be used for other types of voltage generators that require a similar behavior.

Using the EPICS portable channel access server, we have constructed a gateway to the SCADA-based DMACS system that manages the DØ cryogenic and gas utilities.

## 3 Global Event Reporting

EPICS provides tools for handling alarms generated from PVs, including an operator alarm display. However, alarms from slow controls are neither the only nor, necessarily, the most important events in the DØ data acquisition system (DAQ). To process events from all DAQ sources, of which slow controls is but one, we have developed a separate facility, the Significant Event System (SES) [4], to collect and distribute all changes of state of the detector and the data acquisition system. The SES has a central server that collects event messages from sender clients and filters them, via a Boolean expression, for receiving clients. Sender clients, which include the IOCs, connect to the server via ITC, a locally developed, inter-task communication package based upon TCP/IP sockets. All state changes on those clients, including alarm transitions, are sent to the server. The client-server model has advantages over the EPICS alarm facility, where the operator display explicitly connects to each PV. There is no need to construct extensive configuration files for the hundreds of thousands of PVs in the slow controls system and the savings in connect time at startup can be significant.

The alarm class of SES messages receives special handling in the server. The SES server maintains the current alarm state of the entire detector so that receiving clients are able to obtain the current state when they first connect to the server. In addition to specialized receiving clients that may connect to the server, there are three standard clients: the SES logger, the SES alarm display, and the SES alarm watcher. The logger has a pass-all filter so that it receives all SES messages sent to the server and writes the messages received to a disk file. The current state of the detector stored in the server is relayed to users through the alarm display. There is a global configuration for the alarm display and the ability to specialize the configuration for the purposes of individual sub-detectors. For alarms that compromise data quality the alarm watcher will automatically pause the current run. In addition to its monitoring and logging functions, the SES system provides the means for distributing synchronizing messages to other components of the online software system.

Software tools have been developed for mining data from the SES log files. Hardware experts review the log files to understand which hardware devices are unstable and collaborators performing data analysis can insure the event they are examining is real and not caused by a fault in the detector.

The SES server and most of the receiving clients have been coded in the Python scripting language, while many of the sending clients are coded in C or C++. We anticipate that, for efficiency considerations, the server may require recoding in C++ at some later stage in the development cycle. API's for SES clients are available in all three languages.

## 4 Centralized Device Database

The EPICS databases that configure the individual IOCs are flat, ASCII files of record definitions, the database equivalent of a PV, that are read by the IOCs during startup. The EPICS system additionally provides a higher-level construct, called a template, which is a parameterized collection of record definitions. Generator files, which reference the templates, supply the parameter values to produce instances of these templated devices. While these collections of files are adequate for EPICS initialization, they are not easily accessible to host-level processes that may require the same information.

To address this problem, the DØ experiment centralized the relevant device information in a relational database (Oracle) [5] and provided a family of scripts, written in the Python language, to manage the transformation between the relational database and the EPICS, ASCII-format files. In addition to serving as a repository of the EPICS objects — records, record types, templates and generators — the database also stores information about the front-end IOCs: the physical location as well as the location and type of the devices that reside on the IOC. The database can also accommodate a collection of non-templated EPICS records. At the time that this document was prepared, the database contained ~5700 templated devices, corresponding to ~117000 process variables, and that number is constantly expanding

By providing scripts for bi-directional conversions, it is possible to edit collections of devices (instances of templated devices) by extracting the parameterized devices to a generator file, modifying the generator file with a text editor, and re-inserting the generator file into the relational database. For large collections of devices, this three-stage process is often simpler and faster than using a database editor directly.

In addition to the database management scripts, a WWW browser interface to the relational database is available for the initial definition, modification, and viewing of the relational database entries.

With control system device specifications now centralized in the relational database, they are easily accessible to other host-level processes. This, in turn, has led to a series of extensions to the original database schema to support the needs of other, controls-related processes.

## 5 Detector Configuration Management

One of the most complex tasks performed by the control system is the configuration of the detector for specific run conditions. The set of distinct configurations, both for normal, data-taking and for calibration runs, is very large; and, so, the usual technique of uploading a specific detector configuration, once the required conditions are established, and saving it as a file for subsequent downloading is impractical.

For ease of configuration management, the detector is represented as a tree with nodes at successively deeper levels corresponding to smaller, more specialized organizational units of the detector. The terminal nodes of the tree are instances of the high-level devices discussed in the preceding database section. The intermediate nodes of the tree primarily serve to organize the traversal order of the subordinate nodes since the detector is, in general, sensitive to the order in which devices are initialized. The terminal nodes, called action nodes, manage the configuration of a specific, high-level device.

One level of intermediate node, the geographical sector, has particular significance. These nodes, in most cases, represent the individual read-out crates of the data-acquisition system and are the lowest level in the tree hierarchy in which the sub-trees are guaranteed to be functionally independent. The loading process for these nodes may be executed in parallel, significantly reducing the total time required to configure the detector.

A single server program, COMICS [6], coded in the Python language, manages configuration of the EPICS-accessible part of the detector. The tree nodes, both intermediate and action, are all specialized instances of a base node class that defines the majority of the methods that characterize node behavior. The detector tree structure is defined by a set of configuration files that are Python program segments which instantiate instances of these nodes.

## 6 Operator Interfaces and Applications

The experiment selected two programming languages for developing applications and graphical operator interfaces (GUIs): C++ and Python. By providing the Python scripting language with an interface to the EPICS channel access API (Application Program Interface), members of the DØ collaboration have been able to write nearly all of the operator interfaces to the experiment in a high-level, object-oriented language. The advantages of using Python are: (1) it is fundamentally object oriented, (2) it has a number of high-level language constructs, and (3) it has an extensive library that provides interfaces to standard UNIX and LINUX utilities. As programs written in scripting languages tend to be significantly easier to debug, the development time for building the DØ online system was significantly reduced compared with what would have been required had the C++ language been used instead.

GUIs are written with the Python Tkinter interface to the Tk toolkit and with Python Mega Widgets, which are extensions to the standard Tkinter widget set. Using these two collections of graphics objects, we have developed an application framework to assist in developing operator interfaces and to provide a consistent look and feel for all visual displays. This framework includes a collection of specialized, graphical objects for constructing updating displays of PV values.

The experiment uses more than 40 instances of these monitoring displays in the control room to manage the detector components.

## 7 Secondary Data Acquisition

Although originally intended to serve as a short-term replacement for the primary data acquisition system (PDAQ) during the commissioning phase of the experiment, the secondary data acquisition system (SDAQ) is now used for a variety of run-time tasks that includes the calibration and monitoring of detector systems. In place of the dedicated high-speed network used by PDAQ, SDAQ uses a DØ product, ITC, to transmit data packets, gathered on an IOC, to host-based processes. The detector-specific components of a SDAQ application have access to a library of SDAQ functions that handle queuing of data messages between components, interrupt management with callbacks, run synchronization (start/stop/pause/resume events), and priority-based scheduling. Two of the sub-detectors, SMT (Silicon Microstrip Tracker) and CFT (Central scintillating Fiber Tracker) use the SDAQ system: (1) SMT to monitor the performance of individual silicon detector channels during a run, and (2) CFT to calibrate the response of the fiber tracking channels.

## 8 Archiving EPICS Data

While extensively using EPICS records for control and monitoring tasks, almost every detector group in DØ needs to keep and have structured access to archived PV values. There are two major archiving tools employed by D0: (1) the Channel Archiver [7], for detector group specific needs which require rather fast sampling rates, immediate analysis, but does not require frequent access to the old historical data; and (2) the EPICS/Oracle Archiver [8], for long-term studies which require slow sampling rates, easy access to data at any moment, and minimal maintenance.

Many different Channel Archivers are running all the time, writing several thousand PV values. About once a week collected archives are sent to the central Fermilab robotic tape storage via the SAM data management system [9]. The Channel Archiver toolset has many user-friendly interfaces, including web-based tools, which allow data retrieval from an archive in different formats to generate time plots with various options.

Specifically developed to store slowly changing data directly in the Oracle database, the EPICS Oracle Archiver, written in Python and running as a collection of time-scheduled jobs, stores about 1500 PVs with sampling rates varying from 1 minute to 1 hour. Several user interfaces exist to provide an efficient way to extract and plot archived data.

## 9 ACNET Gateway

For operating the D0 detector, it is vital to have a fast and reliable messaging connection between D0 and accelerator operations to exchange control and monitoring information. EPICS supplies cryogenic and magnet data and forward proton detector pot positions and counter rates. ACNET, in turn, sends information about critical accelerator devices. To provide this interchange, we have developed a gateway between the EPICS-based DØ control system and the accelerator ACNET control system. It is implemented as a multithreaded application in Python, using the XML-RPC server/client model, with embedded EPICS channel access and ITC interfaces for D0 clients.

## 10 References

[1] L. R. Dalesio et al., “The Experimental Physics and Industrial Control System Architecture: Past, Present, and Future”, Proc. ICALEPCS, Berlin, Germany, 1993, pp 179-184

[2] J. F. Bartlett, “DØ High Voltage System Tutorial”, DØ Note 4369

[3] D. Harel, “Statecharts: A Visual Formalism for Complex Systems”, Science of Computer Programming, 8 (1987) 231-274

[4] G. Savage, “Significant Event System Tutorial”, Internal DØ document

[5] S. Krzywdzinski, “EPICS Oracle Database Tutorial”, Internal DØ document

[6] J. F. Bartlett, “COMICS: DØ Detector Download Tutorial”, Internal DØ document

[7] K.Kazemir, L.R.Dalesio, “Data Archiving in EPICS”, ICALEPCS99, Trieste, 1999

[8] V.Sirotenko, “Oracle EPICS Archiver”, Internal DØ document

[9] I.Terekhov et al., “Distributed Data Access and Resource Management in the DØ SAM System”, HPDC 2001, 87